GIGA+: Scalable Directories for Shared File Systems

Swapnil V. Patil and Garth Gibson
Carnegie Mellon University

www.google.com/events/scalability_seattle
www.youtube.com/watch?v=2N36SE2T48Q

Use cases for huge directories

- Apps use FS as fast, lightweight database
  - Use case: All clients inserting millions of small files in a single directory as fast as possible
  - Retain VFS API: create(), lookup(), readdir(), etc.
- Creating many small files in a "burst"
  - E.g., per-process checkpoint on large clusters
  - E.g., science experimental capture
- Creating many small files "steadily"
  - E.g., "log" files from long-running apps for later post-processing (history, bio device runs,…)
- Most interested in pushing the boundaries

GIGA+ directory index

- POSIX-compliant file system directories
  - Extreme scalability through high parallelism
  - No range queries

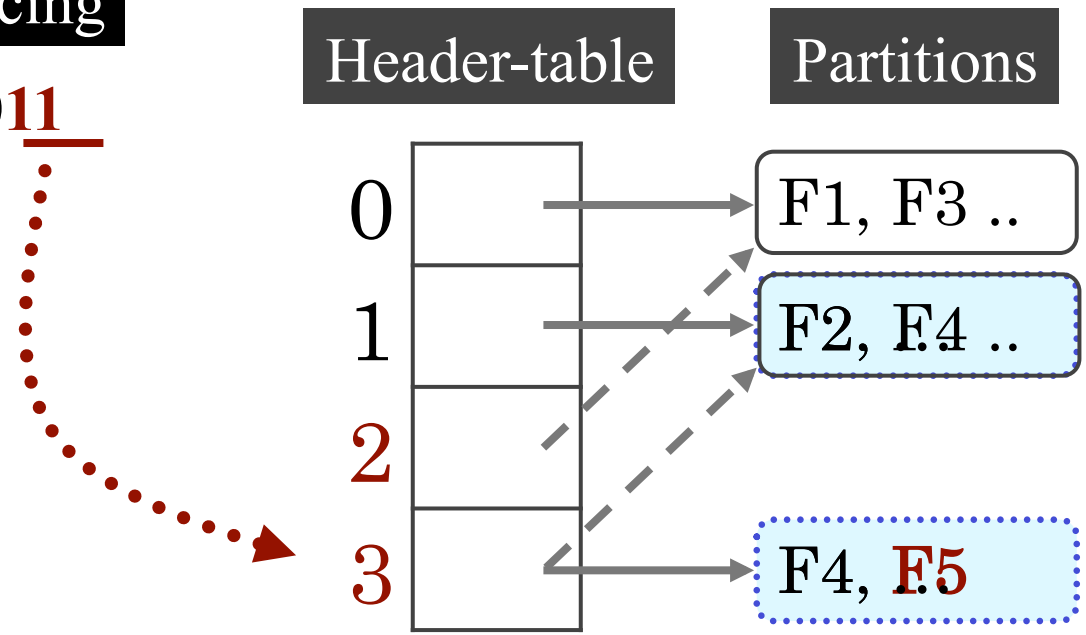
- GIGA+ distributed indexing technique
  - Unsynchronized, parallel growth without any central coordinator
    - Incremental, load-balanced growth
  - Tolerates stale mapping information at the clients
  - Self-describing bitmap to encode the entire index

Extendible Hashing [Fagin79]

Hash keys for load-balancing

hash("F5") = 1001…011

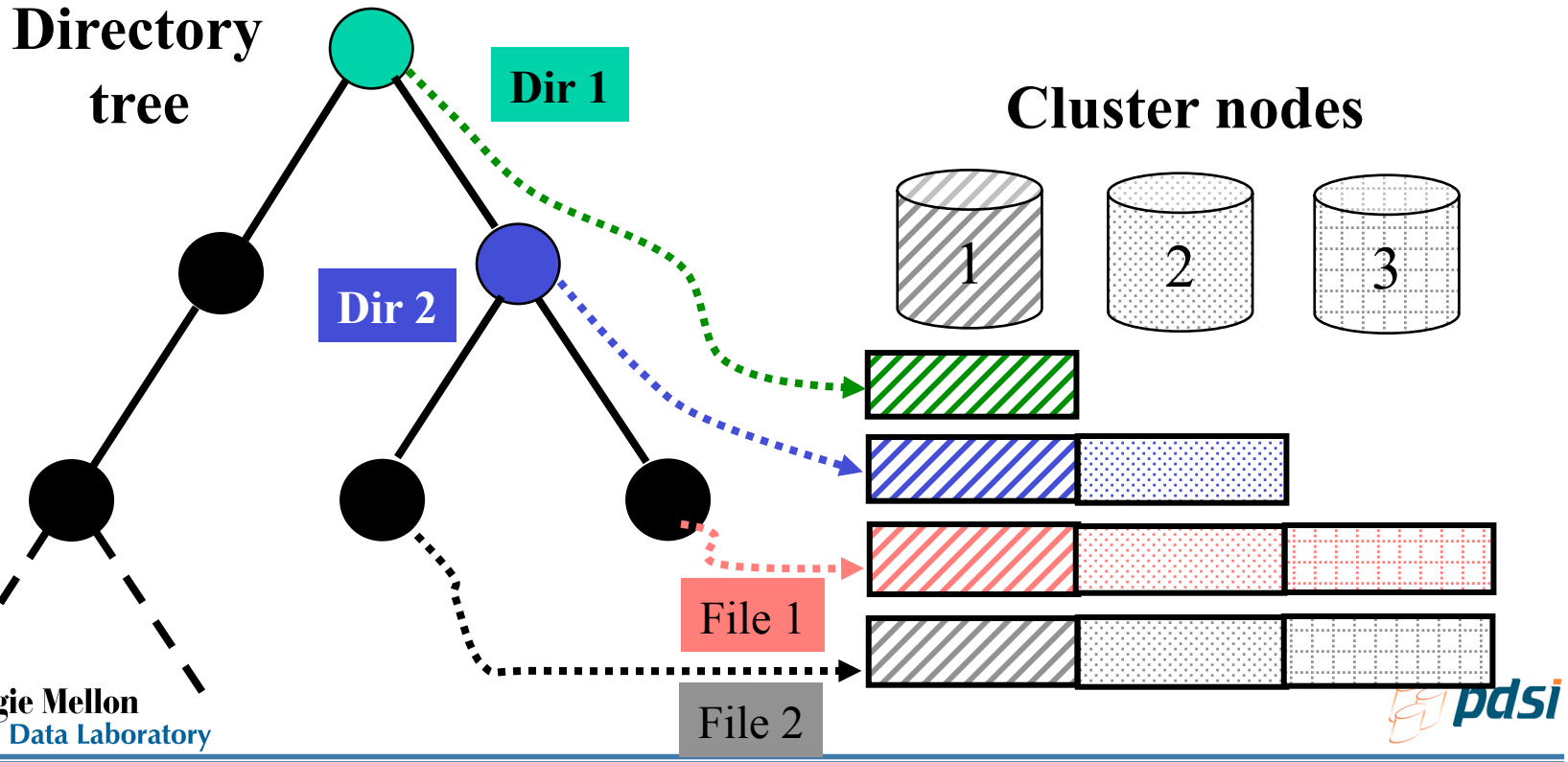
RADIX increases, that uses the growing table (R = 2 bits)



- Header-table doubles, if necessary
  - On splitting, the new partitions distribute their keys
- Mechanism designed for single server impln.

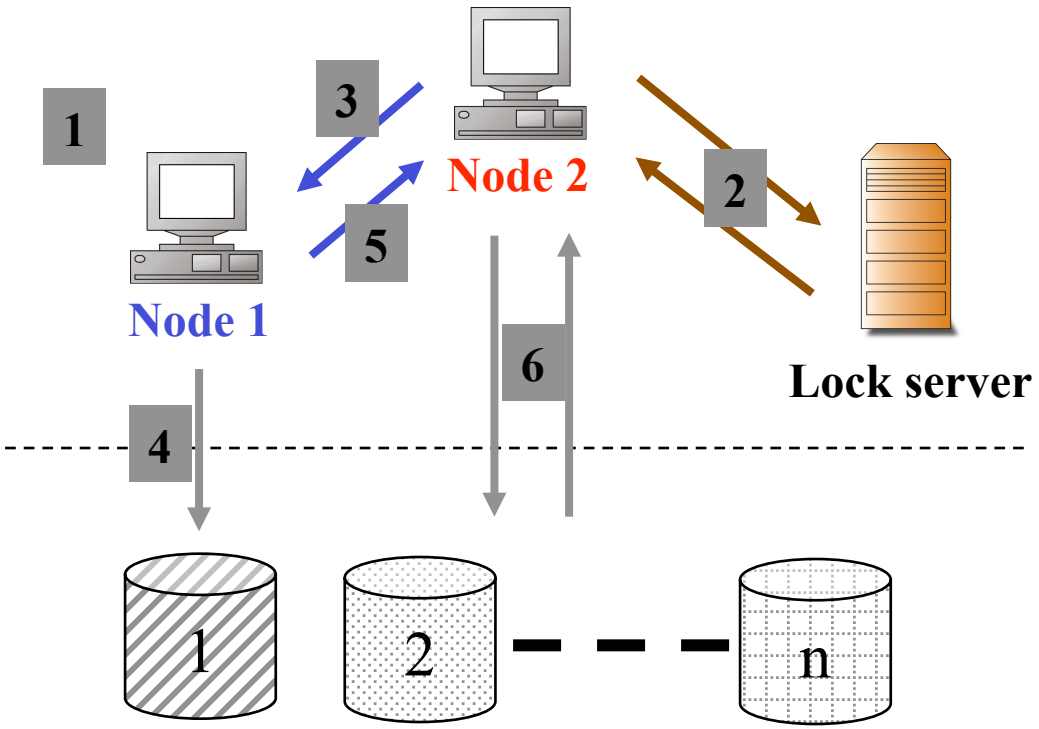
Today: Dirs in IBM GPFS [Schmuck02]

- Distributed directories use extendible hashing [Fagin79], with locking and cache consistency



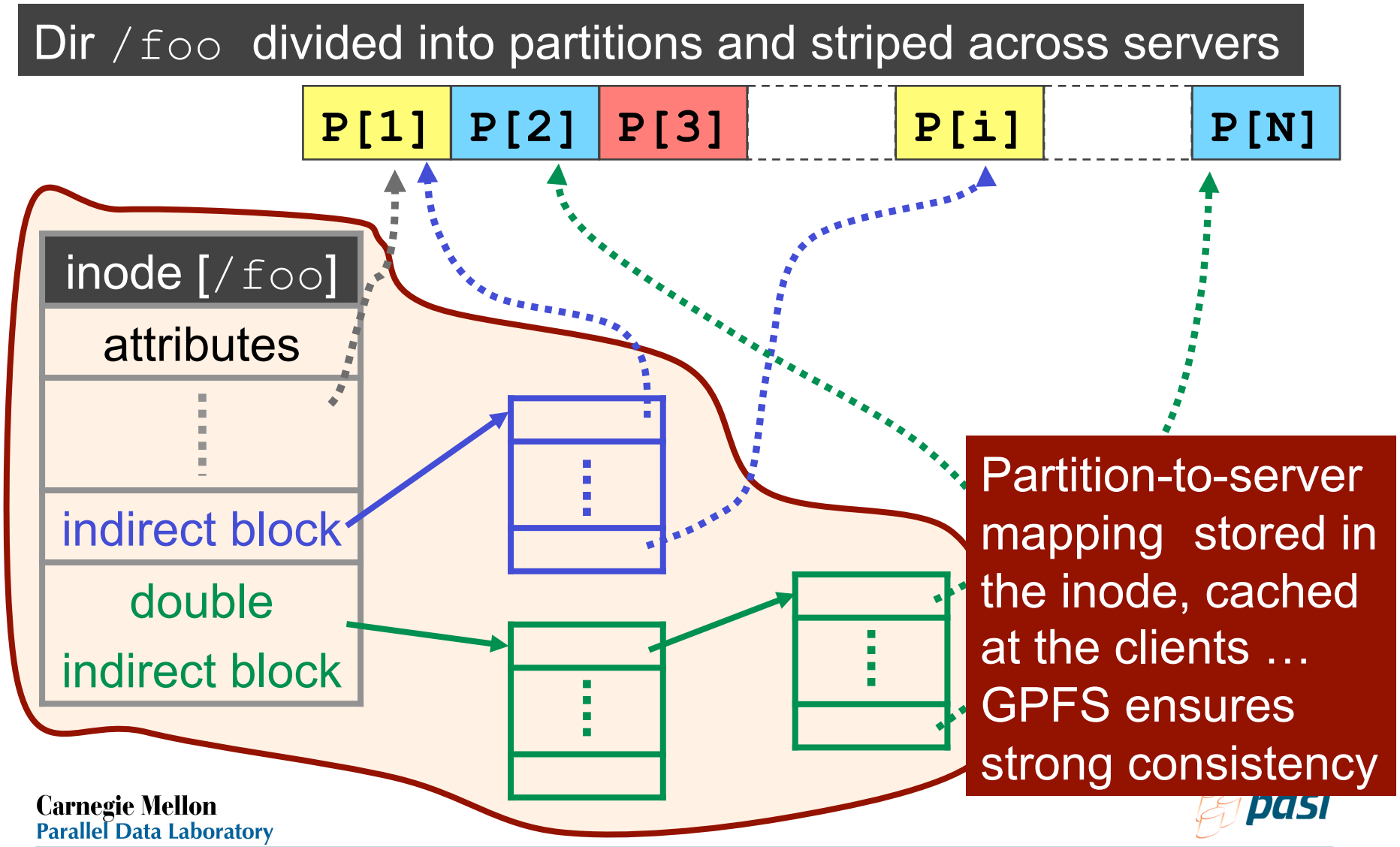
Concurrent inserts in GPFS

- Uses distributed locking and strong consistency (will get better soon!)



1	Node 1 has a write lock on a partition of a directory
2	Node 2 needs to access the same partition and contacts the lock server.
3	Node 2 contacts Node 1 for the write lock
4	Node 1 flushes its cache by writing the partition to disk
5	Node 1 gives the write lock to Node 2
6	Node 2 reads the partition from the disk into its memory

Future bottleneck: map consistency



Reaching for more scaling

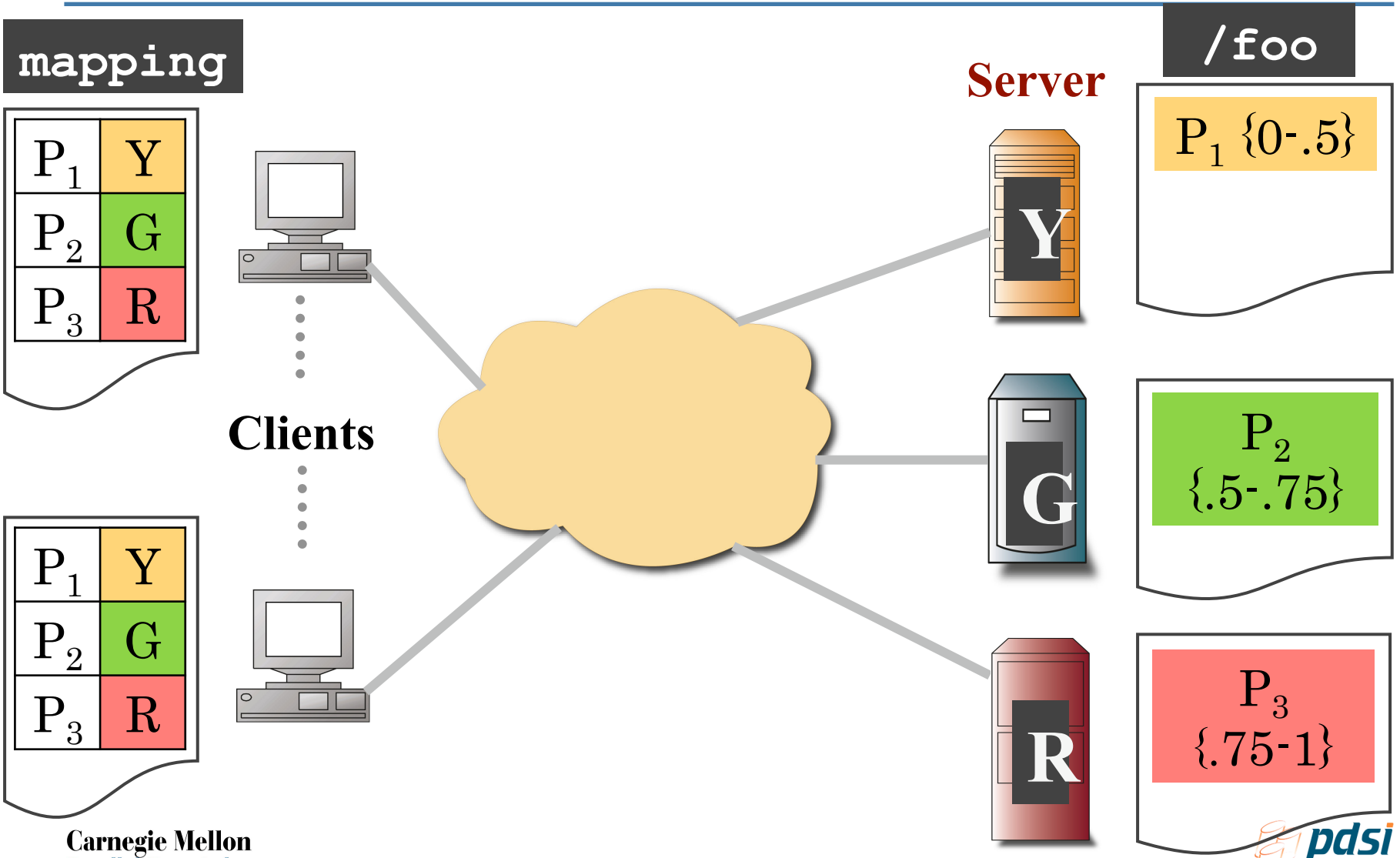
- No need to lookup partition-to-server mapping
  - Use a mapping that is known a priori
  - Use the index size to find which partition to insert
- Tolerate stale mapping information
  - Servers verify cached state and then forward (and correct) client requests to the right server

- LH* [Litwin96] enables these properties but …
  - Imposes a strictly serialized order of splitting
  - No parallelism: only splits one partition at a time

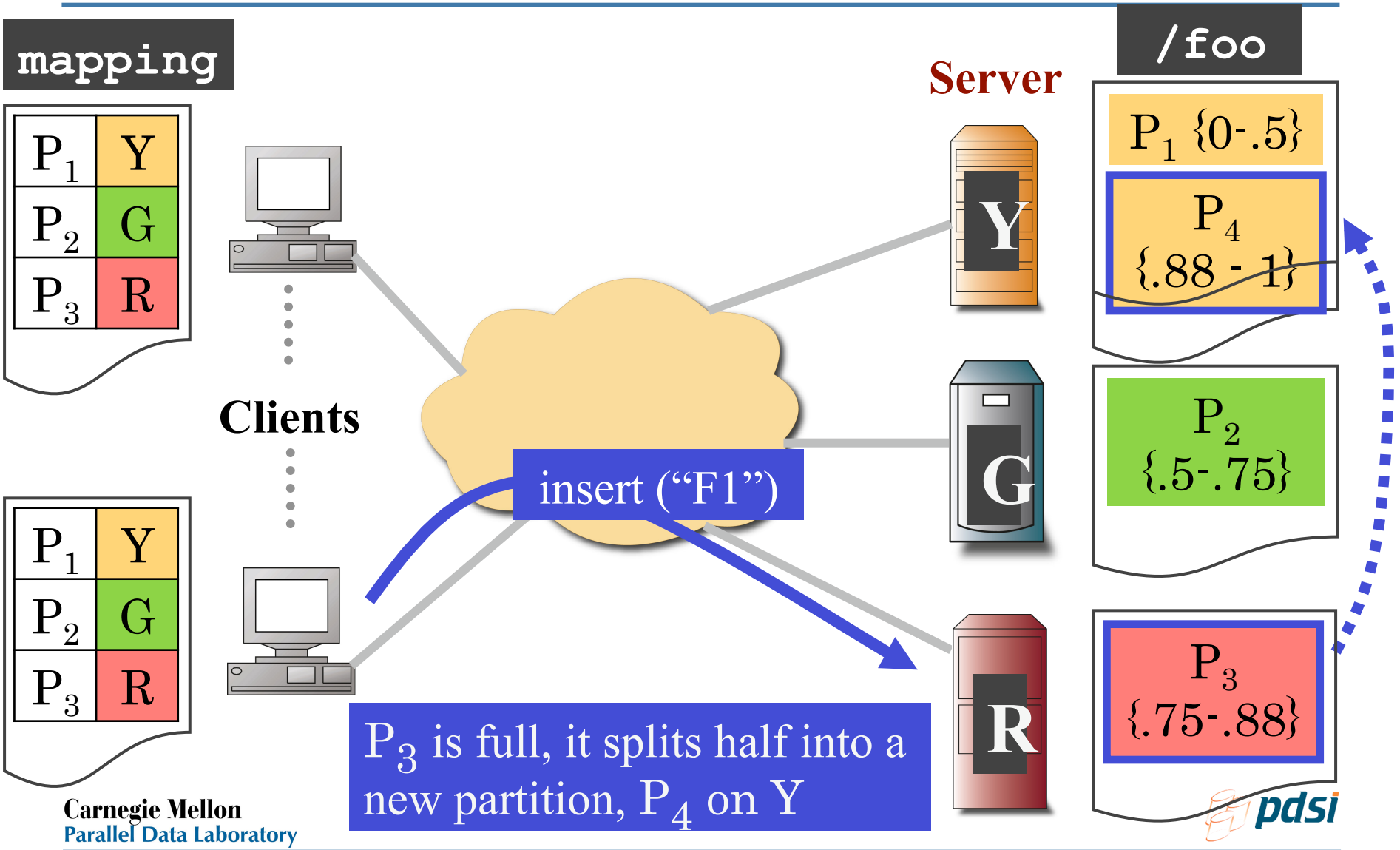
What's new in GIGA+ directories?

- Eliminate serialization
  - All servers grow the directory independently, in parallel, without any co-ordinator
- No synchronization & consistency bottlenecks
  - Servers only keep local "view", no shared state
- Weak consistency of mapping
  - Tolerates the use of stale mapping state at clients
  - Apps and users see strong consistency
    - Once a file is created, lookups can see it

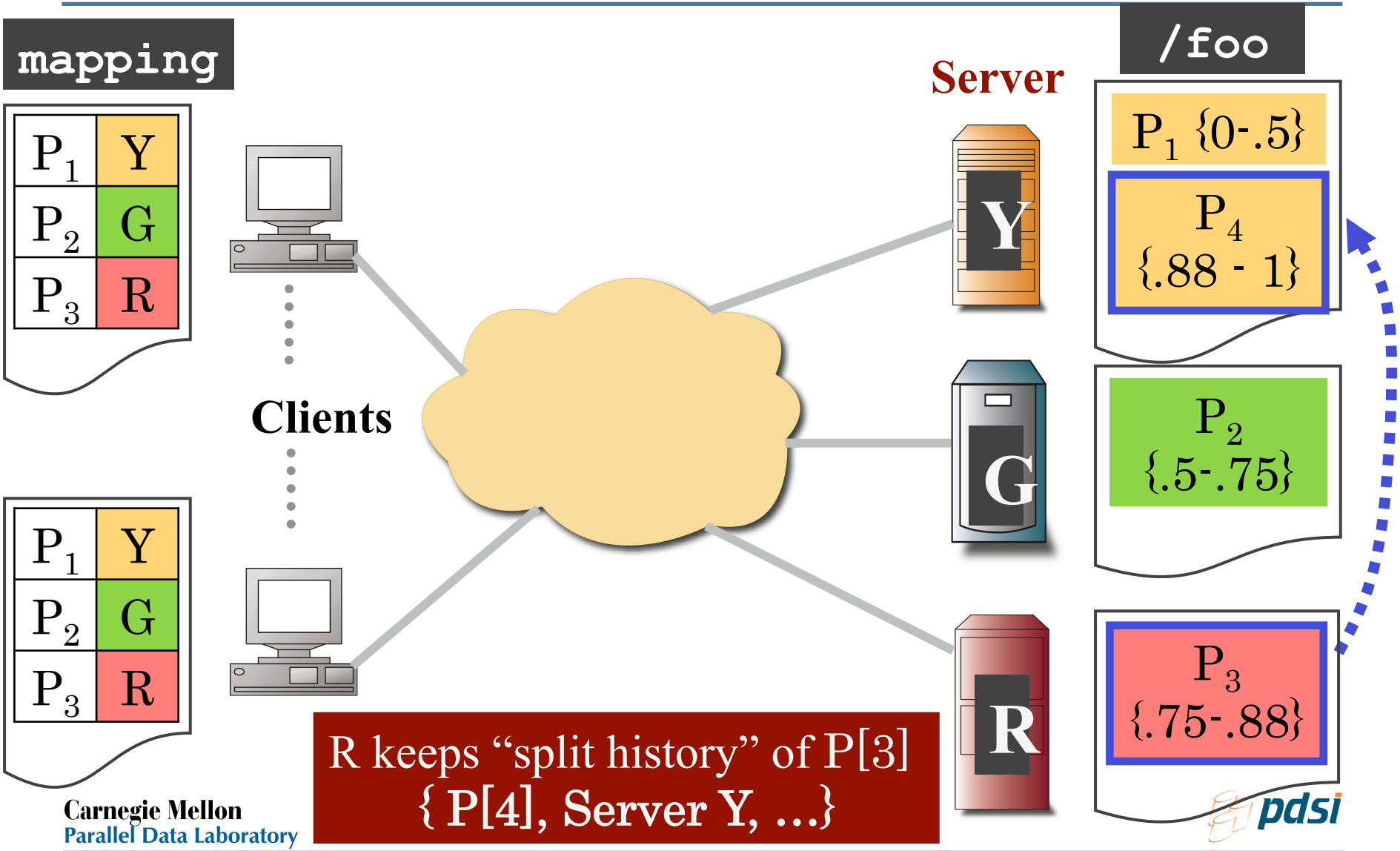
GIGA+ in action



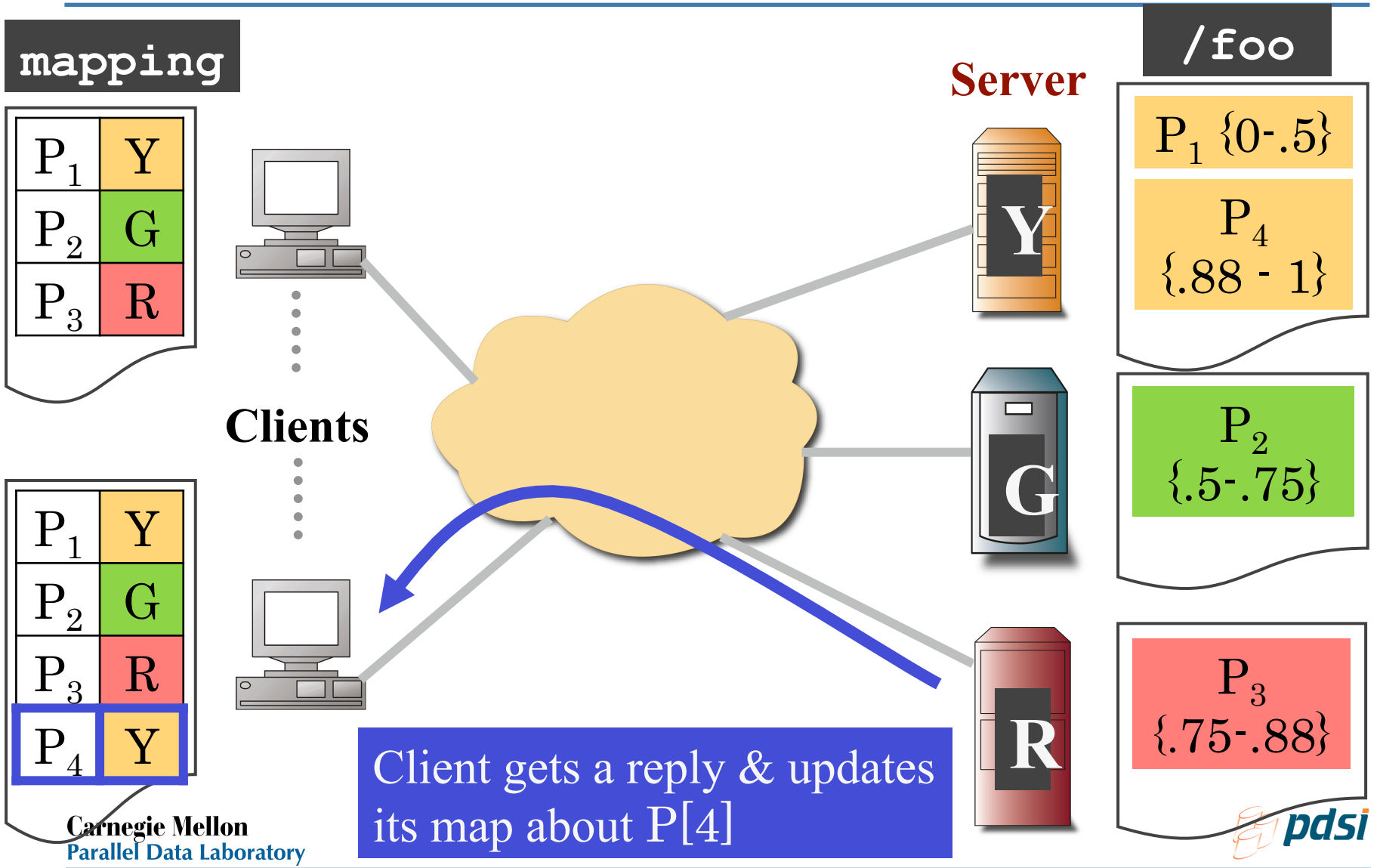
GIGA+ in action



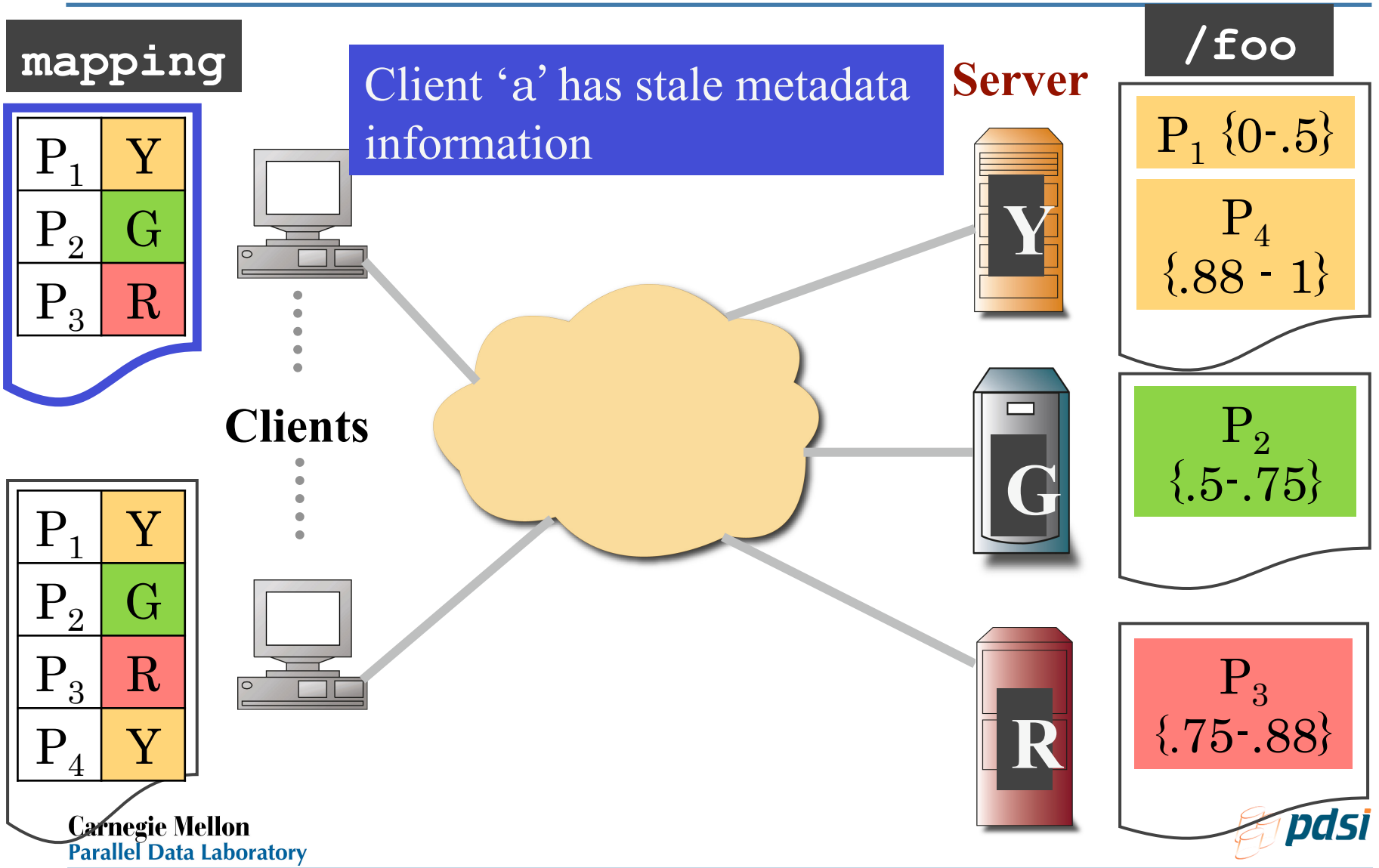
GIGA+ in action



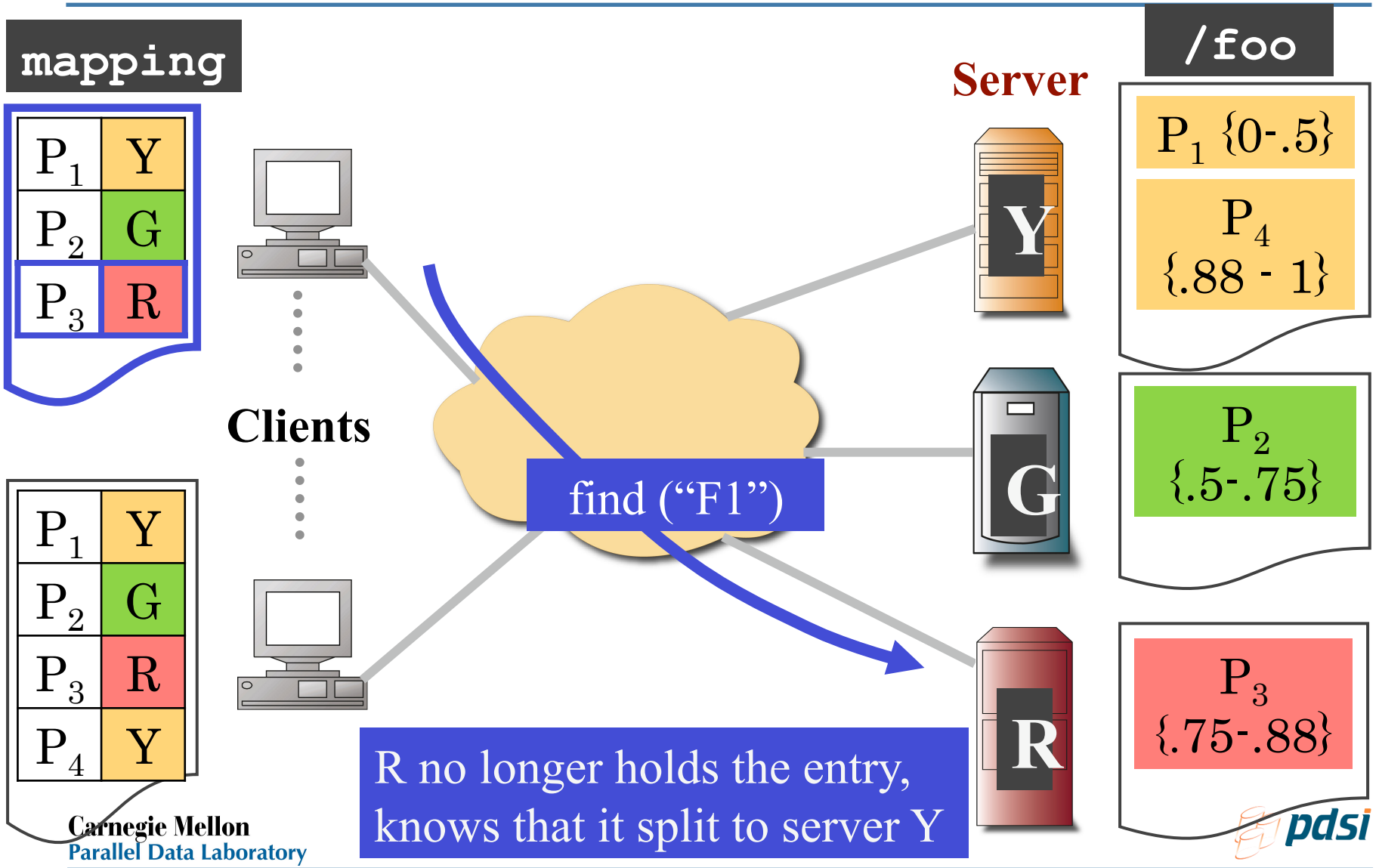
GIGA+ in action



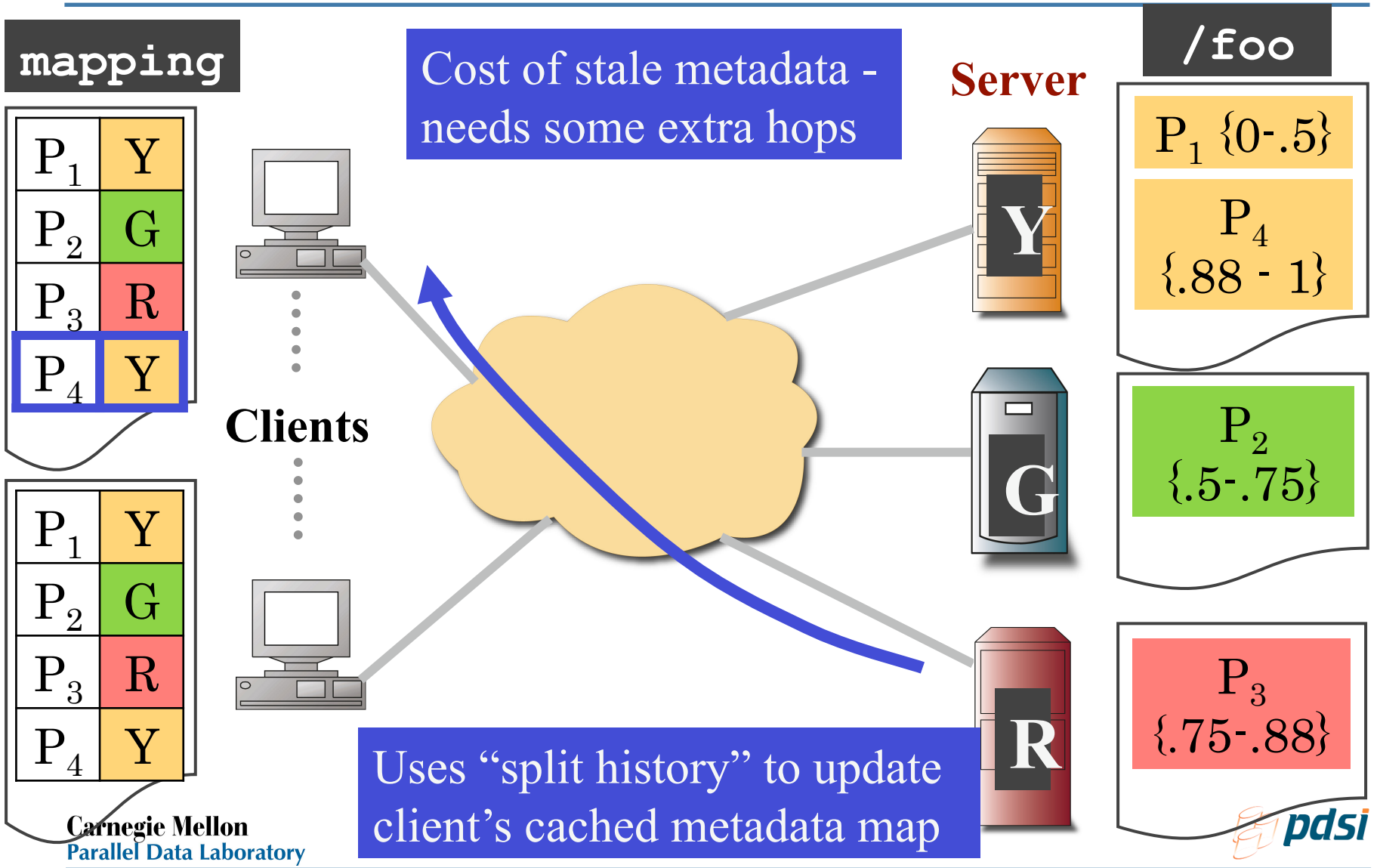
GIGA+ in action



GIGA+ in action



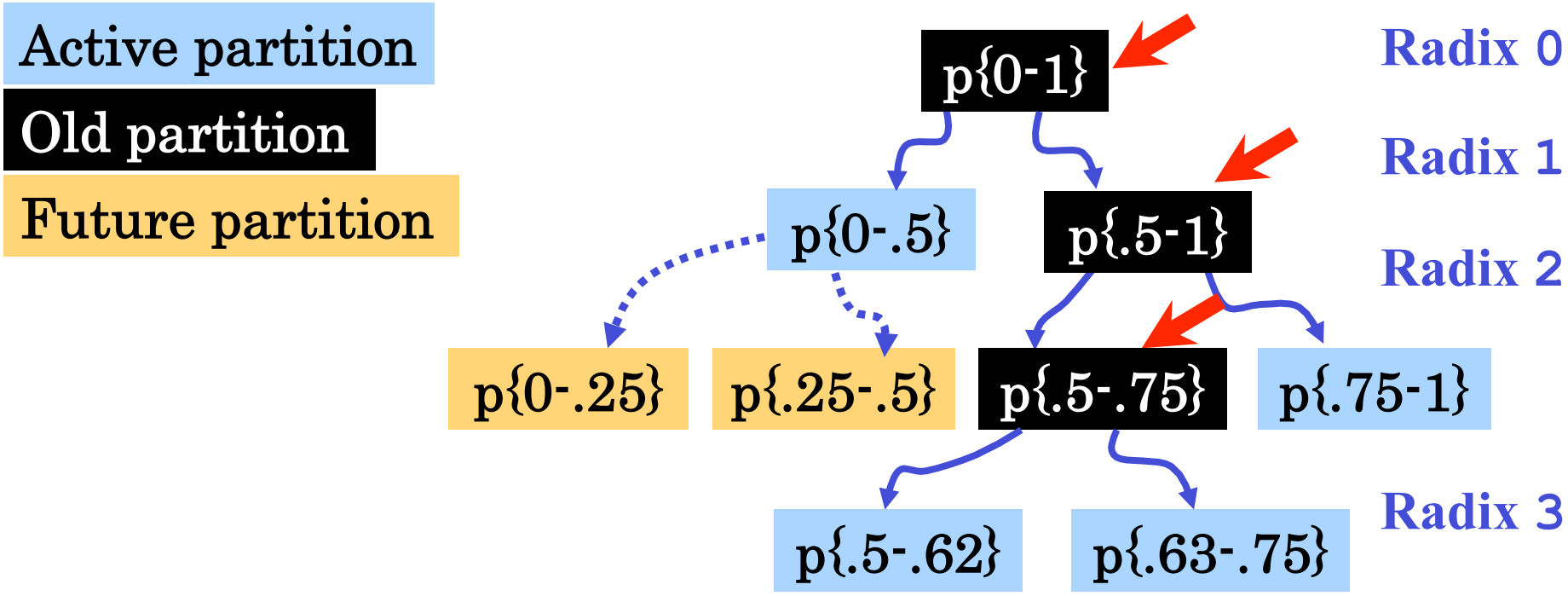
GIGA+ in action



Keeping track of partitions

- Self-describing bitmap for the entire index
  - Indicates the “presence” or “absence” of a partition
- Servers keep track of their partitions
  - Only keep local, current state of partitions
  - Bitmap used to provide lookup hints for the clients
- Clients uses it to lookup a partition
  - Merges (OR operation) bitmaps from diff servers
  - Complete bitmap gives an approximate map of all partitions on all servers

Growth of the directory index



- Each server splits its partition when the partition is full, without telling other servers

Concurrent growth of GIGA+ index

- Fast, concurrent growth through minimal synchronization
  - Servers decide independently when to split partitions
    - Only keep track of their partitions
    - No globally shared state on the servers
  - Servers don't sync with the rest of the system

- Servers keep a split history of its partitions
  - Edges pointing to the children nodes in the tree
  - Used to correct the clients with stale mappings

GIGA+ Design Summary

- Completely decentralized and parallel growth by allowing servers to split independently
  - Each server splits a partition when it wants, without synchronizing with the rest of the system

- Indexing technique that allows use stale metadata mapping at clients
  - Servers update clients' mapping information using bitmaps

Acknowledgements

- Several people involved at some point …
  - S.Lang and R.Ross (Argonne National Lab)
    - PVFS2 prototype co-conspirators
  - S.Hase, A.Jayaraman, V.Perneti, S.Sundaraman
    - Initial FUSE prototype class project
  - M.Polte, G.Ganger, C.Faloutsos
    - General discussions and feedback